A Sketch of a Deep Learning Approach for Discovering UML Class Diagrams from System's Textual Specification

Yves Rigou
Département de mathématiques, d'informatique et de génie
UQAR
Rimouski, Canada
yves.rigou@uqar.ca

Dany Lamontagne
Département de mathématiques, d'informatique et de génie
UQAR
Rimouski, Canada
dany.lamontagne@uqar.ca

Ismaïl Khriss
Département de mathématiques, d'informatique et de génie
UQAR
Rimouski, Canada
ismail_khriss@uqar.ca

***Abstract*—Drafting a formal or semi-formal model describing the functional requirements of a system from a textual specification is a prerequisite in the context of a model-driven engineering approach, such as the model-driven architecture initiative proposed by OMG. This model, called a platform-independent model (PIM), is used to derive automatically or semi-automatically the source code of a system. Different knowledge-based approaches have been proposed to extract a PIM from a textual specification automatically. These approaches use a predefined set of rules to perform this discovery. These approaches impose several restrictions on the way a specification is written. The emergence of machine learning techniques and more specifically of deep learning and their obvious success among others in several tasks in automatic language processing, such as speech recognition and translation, suggests the possibility of using these techniques to reach our objective. In this paper, we review state of the art in the domain and we sketch a rough deep learning approach to achieve our objective of extracting a PIM from the textual specification of a system.**

Keywords—Natural language processing, deep learning, recurrent neural networks, Platform independent model extraction, UML class diagram, textual specification.

I. INTRODUCTION

Implementing a software solution to meet business needs is a complex process that involves several steps. The first step is to translate the needs into the requirements of the future system, usually in a specification written in natural language. In the context of a model-driven engineering approach, such as the model-driven architecture initiative proposed by OMG [1], the second step is to translate these needs into formal or semi-formal language by developing a platform-independent model (PIM). This model can be captured by UML diagrams such as class diagrams for describing the static view of a system. The third step is to adapt this platform-independent model to a platform-dependent model (PSM), obtained by the adoption of the implementation platform of a technology or a set of technologies. Finally, the fourth step is to produce the solution source code.

Once the PIM is set, the process involves only semi-formal languages and is therefore automatable. On the other hand, the extraction of the PIM, ensuring the translation of the requirements of a system into a formal or semi-formal language, is more difficult to process by a machine because of the complexity of natural languages and is therefore, traditionally the responsibility of the human. However, the development of the natural language processing field (NLP) by machines opens up possibilities for automating this step. NLP includes a set of analysis processes that extracts essential information from a text written in natural language. It can be a lexical analysis, a syntactic analysis, a semantic analysis, a speech analysis, etc. This necessary information is then used to develop a PIM.

Different knowledge-based approaches have been proposed to extract a PIM from a textual specification. These approaches use a predefined set of rules to perform this extraction. These approaches impose several restrictions on the way a specification is written. For instance, they support only certain types of grammatical constructions. Moreover, they obtain encouraging but unconvincing results. The models are, at best, an attractive sketch of a real PIM that has yet to be completed by an expert.

The emergence of machine learning techniques and more of deep learning and their obvious success among others in several tasks in automatic language processing such as speech recognition and translation suggest the possibility of using these techniques to reach our objective. The interest of these techniques is to remedy the problems encountered by the approaches based on hard-coded knowledge and instead to let the systems be able to acquire their own knowledge from raw data.

The study of the literature of deep learning in automatic processing led us to identify three communities of researchers working in isolation. Each community has a specific task, specific literature, its own performance competitions, and its own standard datasets. These tasks are named entity recognition (NER), coreference resolution, and relationships classification. We strongly believe that an effective approach to discovering a PIM from a textual specification should try to combine these three tasks.

In this paper, we review in section 2 the state of the art in the domain. In section 3, we sketch a rough approach to achieve our objective of extracting a PIM from a textual specification of a system. Finally, section 4 provides some concluding remarks and discusses our ongoing works.

978-1-7281-4979-0/20/$31.00 ©2020 IEEE

II. RELATED WORK

Several studies have focused on the development of a PIM in an automatic or semi-automatic way [2]. These all make particular choices on some key points that can be categorized into different categories. Firstly, the initial assumptions that combine the representation model of the information used at the input of the system, the level of restriction of the language used in the drafting of the requirements, and the use of complementary modules to clarify the terms of the system business area. Then, the tactical decision that corresponds to the nature of the PIM produced. Finally, the development process that includes the preprocessing stages of natural language and the production of the PIM.

A. Starting assumptions

The first starting assumption concerns the model of representation of the information used at the input of the system. Two main directions emerge. On the one hand, studies that start from a specification drawn up without any particular processing [3, 4, 5, 6], which represents the standard way of work from the industry. The difficulty is that no formalism is integrated into the starting point, which complicates all the machine processing that must be performed later.

On the other hand, studies that choose to impose a predefined formalism taking, for example, the form of a use case model [7, 8, 9] involving actors, a system and requirements linking both expressed in natural language. The disadvantage of the approach is that analysts must adhere to a set of presentation rules to write their requirements, which needs getting familiar with the tool before starting writing. The advantage is that the formalization of requirements starts early, which facilitates the subsequent automatic or semi-automatic processing.

The second assumption concerns the level of restriction of the language used in the drafting of the requirements. Restrictions are a set of rules that are imposed when writing to simplify sentences and improve processing. The simpler and more structured the sentences are, the easier it is to extract the information. This key point is linked to a certain extent to the previous one since studies that choose to impose a formalism [7, 8, 9] implicitly also impose certain restrictions on language. But the restrictions can also concern studies that want to start purely from a natural language while imposing particular structures on the sentences. These restrictions can take several forms: the use of simple syntaxes such as subject, verb, object [4, 9], the systematic use of the active way [4], use of simple subordinate clauses [4], restriction of the use of pronouns [9], etc. Some studies are betting on imposing no restrictions on the language used [3, 6, 10, 11, 12, 13]. One must then ask the question of the coverage of the generated model with the formulated requirements.

The third assumption is the use of complementary modules to clarify either the terms of the business domain or the links between various concepts [2]. The first category consists mainly of glossaries [7]. The second is composed of different models that summarize general knowledge about how objects can interact together [5, 12]. Again, the purpose of such modules is to facilitate the automatic or semi-automatic processing step by providing a layer of additional information. Older studies are more concerned with the use of these modules [3, 10], while more recent studies attempt to extract information in a more naïve and reproducible way from a domain to another by focusing solely on the information contained in the document [6, 9, 13].

B. The nature of the PIM

The final model generated is most often a class diagram [5, 6, 9, 12, 13]. Some studies produce a combination of documents, for example, a domain model, and an activity diagram [11] or a robustness diagram, a communication diagram, and a class diagram [7]. More recent studies produce more complete models than older ones since the techniques have been refined. Class detection has been conclusive for a long time, but the discovery of attributes, methods, and relationships, particularly composition and generalization relations, is more problematic in some cases.

C. The production process

The production process comprises the following stages: a set of preprocessing steps and the production of the PIM.

1) Preprocessing steps: All studies that take a natural language form as a starting point use one or more preprocessing steps that correspond to different kinds of NLP. These steps are carried out chronologically, the result of a step serving as a starting point for the next step, so that the text written in a natural language quickly disappears in favor of more schematic and standardized representations. In this sense, they participate in the progressive formalization of requirements to reduce the variability of natural language and to transpose it into a set of limited structures that are finally apprehended by a set of rules and heuristics to identify the PIM. These steps are morphological/lexical analysis, syntactic analysis, semantic analysis and discourse analysis [14].

a) Morphological/ lexical analysis: Lexical analysis is the entry point of natural language in the process that will lead to the production of the PIM. It plays a fundamental role, and it is, therefore, logical to find it in all studies [3, 5, 6, 7, 11, 12, 13, 15]. The purpose of this step is to separate the sentences from each other to treat them individually as separate units of analysis, to individualize the different words of the document, to identify the basic form if they are conjugated, contracted or plural and finally to give their nature (noun, verb, adjective, adverb, etc.). The original sentences are thus broken down into analysis blocks (the sentences), and each block is broken down into unitary blocks (the words), which constitute the basic elements of the subsequent preprocessing.

b) Syntactic analysis: Syntactic analysis is as typical as lexical analysis [3, 5, 6, 7, 9, 11, 13]. This step is responsible for individually assigning a function to the words (subject, verb, object, etc.). The words are then grouped into atomic groups [6] if they form a coherent entity like a nominal group or a verbal group.

The realization of this step is also entrusted to external modules, often the same as in the case of lexical analysis (parsers). Some modules may add additional information such as a named entity recognition (see section 2) whose use has not been exploited in the context of the automatic or semi-automatic generation of the PIM or a resolution of coreference (see section 2).

The results of lexical and syntactic analyzes can be:

- A syntactic tree [3, 6, 7, 13].
- A list of dependencies [4, 6, 9].

c) Semantic analysis: The semantic analysis takes as a starting point the results of the syntactic analysis carried out at the level of individual words or groups of words to constitute the functional architecture of the sentence. The sentence is broken down into functional groups linked together by dependency relationships. The processing is no longer considered at the level of the individual words but at the level of the coherent groups that compose a sentence. We find ourselves talking about subject groups, and object groups, organized around a verbal group, which is the center of the predicate linking these elements. It's really about who does what and how.

Unlike the previous two steps, this step does not use an external module but is an original contribution of the various studies. The extraction of dependencies between functional groups is achieved by the implementation of a set of rules and simple heuristics [3] or more complex [6, 9, 11], applied to the units extracted from the parsing and specific to each study.

The results of the semantic analysis can be:

- A tabular representation of sentences is a schematic table representation of the various functional components (subject, predicate, object) of sentences [11].
- A graphical representation of the relationships between words is where all the relationships between words in a sentence are schematized graphically [11].
- A list of relationships between words is where information on unit numbers, time, and the verb path are included [3, 6].
- A list of sentence structures [9].

d) Discourse analysis: The analysis of the discourse changes the scale of analysis to relate the information on the objects not only at the level of the sentence but at the level of the whole text. As a result, the information extracted by the individual semantic analysis of the sentences is taken sequentially to collect all the information in a single model [3]. The information is synthesized so that each entity is presented only once but with all its relationships [11].

This implies being able to solve multiple references to the same object throughout the speech. When the denomination is shared, the task is simplified, but it becomes much more complicated when an object is designated by a pronoun. It is then necessary to be able to solve large-scale co-conferences [3], which is still challenging to do. For this reason, discourse analysis as a preprocessing step is uncommon and simplified in cases where it is undertaken.

The results of the discourse analysis can be:

- Semantic networks implemented by [15] and by [11], which are a form of discourse analysis that presents in the form of a graphical model that exhibits the set of relationships that exist between objects within a text. It summarizes all dependencies extracted at the level of individual sentences. This model already approaches a lot of a class diagram that is eventually produced as a PIM.
- A discourse model [3] is a variant of semantic networks and also represents information extracted in a graphical form close to a class diagram.

2) The production of the PIM: The production stage of the PIM is the last phase of the process. It is always based on the application of rules on one or more models produced by the preprocessing steps. The number of preprocessing steps before the production stage of the PIM depends on the studies. The rules translate decisions made, a priori, to determine whether a word or group of words should be part of the model and how it should be represented. They are, therefore, a reflection of what the authors consider, a priori, as the general way of expressing a relation between entities. In the case of a class diagram, which is the most common PIM, it is necessary to determine which expressions must become classes, attributes, methods, which relations must become simple associations, generalizations, compositions or aggregations. The precision of the rules is the key element that makes it possible to produce a complete PIM.

In the shortest paths, the rules are applied to the results of lexical analysis [12] or syntactic analysis [4, 5, 7]. These are rather direct rules based on simple structures like subject-verb-object. In longer pathways, the rules are applied to the results of the semantic analysis [6, 9,13]. Intermediate models produced by semantic analysis present a more complete level of information retrieval. The rules that apply take into account more complex sentence structures. Generally, a final validation step must take place to refine the model and remove any redundancies that may occur when the same entity is considered in several classes. This refinement is done by hand [3, 4] or automatically with another set of rules [7].

III. TOWARDS A DEEP LEARNING APPROACH

In this section, we sketch a rough approach to achieve our objective. We first review our objective, the methodology adopted and we present our initial findings.

A. Objective and methodology

Our aim is to produce a view of a PIM, expressed as a UML class diagram, from the textual specification of a system. This textual specification should use the fewest possible restrictions and specificities.

In concrete terms, the system must take an input document written in a natural language and be able to identify classes, attributes, methods, and relationships between these classes.

As seen in section 2, different knowledge-based approaches have been proposed to extract a PIM from a textual specification. These approaches impose several restrictions on the way a specification is written. Moreover, they obtain encouraging but unconvincing results. The models are, at best, an attractive sketch of a real PIM that has yet to be completed by an expert.

The emergence of deep learning and its evident success, among others in several tasks in automatic language processing such as speech recognition and translation, suggest the possibility of using these techniques to reach our objective.

We started by asking a few questions: is there work in deep learning that is tackling the problem? If not, what are the works approaching us?

Subsequently, we must first make a small proof concept to familiarize ourselves with the recommended architectures in this kind of task as well as the technologies to implement them.

Our study of work in deep learning will allow us to design our system that meets our purpose. For this and like any project in deep learning, it is necessary to follow the practical methodology as recommended by the literature [16]. It is composed of the following steps:

1. Choice of metrics

2. Building an end-to-end system

3. Refinement based on data.

A wide range of metrics are offered depending on the learning task to be accomplished: accuracy, precision, recall, F1 score, etc.

The construction of an end-to-end system makes it possible to be operational as soon as possible. It is recommended to start with the most straightforward viable system. The starting point for starting is to copy the state of the art of a related publication [16]. In this step, we must choose the architecture family suited for our objective. The logical choice for this kind of task is the family of sequential models, such as recurrent neural networks.

Subsequently, we enter the data-driven refinement stage. The goal is to arrive at a system that responds well to the training dataset while allowing generalization to the test set.

B. *Initial findings*

The study of the literature of deep learning in automatic processing led us to identify three communities of researchers working in isolation. Each community has a specific task, specific literature, its own performance competitions, and its own standard datasets. These tasks are named entity recognition (NER), coreference resolution (CoRef), and relationships classification (RC).

Before giving an overview of these tasks, we start by introducing some architectures in deep learning that we mention in the rest of this paper.

a) Overview of some deep learning architectures: The most used architectures for handling text-based tasks are recurrent neural networks (RNNs). Sometimes, convolutional neural networks (CNNs) are also used.

An RNN is a network of a specialized neural network to process sequences of variable lengths and can evolve into very long sequences [16]. RNNs take advantage of one of the first ideas found in machine learning and statistical models of the 1980s: sharing parameters between different parts of a model. Parameter sharing lets you extend and apply the model to examples of different shapes (different lengths, here) and generalize across them [16]. This sharing is particularly essential when specific information may appear in several places in the sequence [16].

We also have the notion of bi-directional RNN (bi-RNN), which combines an RNN that is advancing in time from the beginning of the sequence and another RNN that goes back in time from the end of the sequence [16]. In cases where we need more context, the gap between the relevant information and where it is needed may be huge. For this, the use of networks that allow keeping this memory is necessary. The most used models for this is the LSTM model (long short-term memory) or GRU (gated recurrent unit) [16].

The attention model is proposed in [17] as an extension to the encoder-decoder model in the translation task. The authors explain the model as follows: "*Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.*" [17]. In discourse analysis, the model is used to identify different degrees of importance of words inside a discourse [18].

CNN is typically used to support tasks containing images or photos [16]. It also has this ability to allow the sharing of parameters like RNN; therefore, is sometimes uses in support of sequence inputs.

b) Overview of the three tasks: We first give some definitions:

- Entity: an element of interest that designates an individual, a thing, or a tangible concept that will compose an element of a class diagram.
- Mention: a word or expression of interest in the text that designates an entity, it may be a proper name or a common name.
- Set or cluster: A set of mentions that represents the same entity.

The objective of the NER task is to identify words and expressions of interest. Those depend on the purpose of the study. Generally, we also want to categorize the nature of the entity (Person, Organization, Location, etc.). A neural network is trained on a set of sentences whose words are annotated with a set of tags, usually BIO or BILOU following the studies (B: Begin, I: Inside, O: Outside, L: Last, U: Unit). The principle is relatively constant between studies: the words are transformed into embeddings, then treated by one or more layers of neurons. Embeddings represent the encoding of a word into a vector of real values that represents the semantic proximity of words. The training of these embeddings can be part of the study, but it is also possible to use pre-trained emebddings such as GLoVe [19] or word2vec [20]. A function of type softmax identifies from the output vector the most likely tag for the word considered from the list of available tags. The cost function used is usually a cross-entropy categorical function. Once trained, the network can detect mentions of interests.

Fig. 1 shows an example of tagging two sentences BILOU tags. Note that numbers (1, 2, …) and unit words surrounded by characters '<' and '>' are not used in this task but are here for illustrating the CoRef and RC tasks.

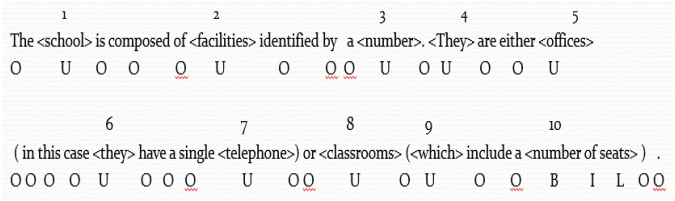

Fig. 1. Example two sentences with tagging

Different standardized training and test databases are used:

- CoNLL 2002 [21], CoNLL 2003 [22] (Conference on
- Computational Natural Language Learning)
- MUC-6 [1], MUC-7 [2] (Message Understanding Conference)
- ACE 2004 [3], ACE 2005 [4] (Automatic Content Extraction)
- GENIA[5]

The most recent architectural models used for this task are bidirectional LSTM or GRU, as in [23, 24, 25].

The CoRef task seeks to find references to the same entity in a text. It is easy for a human but particularly complicated for a machine. Difficulties come from the fact that a machine has to determine if a word or group of words is the first occurrence of an entity or if it is a reference. The list of the words of interest are numbered in the order in which they appear in the text and for each one, it is a question of knowing if a mention already met could be an antecedent. The result is a set of clusters where each of them contains the first mention of interest with all its co-references.

For training words or group of words are identified (for instance, surrounded by the caracters '<' and '>' as in Fig. 1). For the example in the figure, the clusters are : {1}, {2, 4}, {3}, {5, 6}, {7}, {8, 9} and {10}.

Different standardized training and test databases are used:

- CoNLL 2011 [26], CoNLL 2012 [27]
- MUC-6, MUC-7
- ACE 2002[6], ACE 2003[7], ACE 2004, ACE 2005

Here again, the most used architectural model for this tsk is bidirectional LSTM such as in [28] and [29].

The RC task is about finding the relationship (and identifying or classifying the relationship) that exists between two entities of a text. The principle resembles that of the NER since it is a question of assigning a class to a pair of words among a set of predefined classes. The input is not the vector associated with a word but the vector associated with a couple of words. The context of the surrounding words can be used to compose the vector. A function of type softmax is used to assign the category of relations to the couple of mentions and a loss function of type categorical cross-entropy is used for the training. Once the network is trained, it must allow classifying the relationship between two mentions of interests.

As for other tasks, bidirectional LSTM is the most used model in RC, such as in [30] and [31].

SemEval10[8] and KBP37 [32] are standardized training and test databases for this task.

Each dataset proposed its own classification of relationships and how its data is tagged. An example of a relationship is the Whole-Component relationship. For our case in Fig. 1, we have a Whole-Component relationship between the mentions `School` and `Facilities`.

Note that attention mechanisms are often used for CoRef (such as in [29]) and RC (such as in [33]) tasks.

c) Sketch of our deep learning approach for UML class diagram extraction: Our task of extracting class diagrams from the text merges the three tasks: NER, CoRef, and RC. The extraction task is then processed in three steps:

1. Mention detection: this step is an extension of the NER task except that it is not applied only to named entities but rather to nominal and pronominal terms. This must allow the words or expressions that are of interest to come out.

2. Coreference detection: the use of the Coref task above the results of the first step will make it possible to link the multiple references of the same entity.

3. Relationships classification: The third step is an extension of the RC task and, more specifically, the classification of relations. In particular, we are interested in identifying the following relationships:

- class - attribute
- parent class - child class
- composite class - component class
- aggregated class - aggregating class
- class - class.

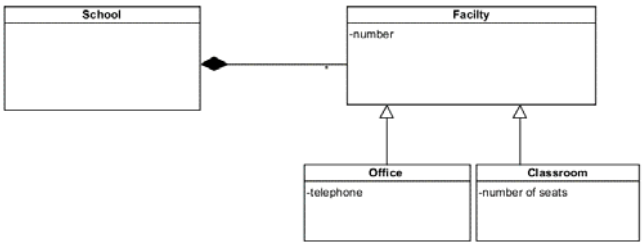

Fig. 2. UML class diagram expressing the two sentences of Fig.1

Note that the datasets used in the three tasks are not adequate for our task, which ask us to develop our own dataset.

Fig.2 gives the class diagram that could be extracted from the two sentences of Fig.1.

[1] https://catalog.ldc.upenn.edu/LDC2003T13 (2019-12-03)

[2] https://catalog.ldc.upenn.edu/LDC2001T02 (2019-12-03)

[3] https://catalog.ldc.upenn.edu/LDC2005T09 (2019-12-03)

[4] https://catalog.ldc.upenn.edu/LDC2006T06 (2019-12-03)

[5] https://orbit.nlm.nih.gov/browse-repository/dataset/human-annotated/83-genia-corpus 2019-12-03)

[6] https://catalog.ldc.upenn.edu/LDC2003T11 (2019-12-03)

[7] https://catalog.ldc.upenn.edu/LDC2004T09 (2019-12-03)

[8] https://www.aclweb.org/anthology/S10-1006/ (2019-12-03)

IV. CONCLUSION AND FUTURE WORKS

Writing a formal or a semi-formal model called a PIM or platform-independent model, describing the functional requirements of a system from a textual specification is a prerequisite in the context of MDE, such as the MDA initiative proposed by OMG. In this paper, we reviewed state of the art in the domain of PIM discovery of textual specifications. We also sketched a rough deep learning approach to achieve our objective. Our approach is now in the process of implementation as well as the development of a data set comprising a set of specification written in English as well as their models expressed in UML class diagrams.

REFERENCES

[1] J. Miller and J. Mukerji, "MDA Guide Version 1.0.1," Object Management Group omg/2003-06-01, 2003.

[2] T. Yue, L. C. Briand, and Y. Labiche, "A systematic review of transformation approaches between user requirements and analysis models," Requirements Engineering, vol. 16, pp. 75-99, June 01 2011.

[3] H. M. Harmain and R. Gaizauskas, "CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis," Automated Software Engineering, vol. 10, pp. 157-181, April 01 2003.

[4] D. Popescu, S. Rugaber, N. Medvidovic, and D. M. Berry, "Reducing Ambiguities in Requirements Specifications Via Automatically Created Object-Oriented Models," Berlin, Heidelberg, 2008, pp. 103-124.

[5] S. D. a. D. Joshi, D., "Textual Requirement Analysis for UML Diagram Extraction by using NLP," International Journal of Computer Applications, vol. 50, pp. 42-46, July 2012 2012.

[6] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Extracting domain models from natural-language requirements: approach and industrial evaluation," presented at the Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-malo, France, 2016.

[7] D. Liu, K. Subramaniam, B. H. Far, and A. Eberlein, "Automating transition from use-cases to class model," in CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No.03CH37436), 2003, pp. 831-834 vol.2.

[8] T. Yue, "Automatically deriving a uml analysis model from a use case model," Carleton University, 2010.

[9] J. S. Thakur and A. Gupta, "Automatic generation of analysis class diagrams from use case specifications," ArXiv, vol. abs/1708.01796, 2017.

[10] L. Mich and R. Garigliano, "NL-OOPS: A Requirements Analysis Tool Based On Natural Language Processing," 2002.

[11] M. G. Ilieva and O. Ormandjieva, "Models Derived from Automatically Analyzed Textual User Requirements," in Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), 2006, pp. 13-21.

[12] M. Ibrahim and R. Ahmad, "Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques," in 2010 Second International Conference on Computer Research and Development, 2010, pp. 200-204.

[13] S. B. K. Bhagat, P. R.; Kapadnis, N.; Patil, D. S.; and Baheti, M. J., "Class Diagram Extraction Using NLP," in 1st International Conference on Recent Trends in Engineering & Technology, 2012, pp. 125-128.

[14] E. D. Liddy, "Enhanced Text Retrieval Using Natural Language Processing," Bulletin of the American Society for Information Science and Technology, vol. 24, pp. 14-16, 1998.

[15] R. Morgan, R. Garigliano, P. Callaghan, S. Poria, M. Smith, A. Urbanowicz, et al., "UNIVERSITY OF DURHAM: DESCRIPTION OF THE LOLITA SYSTEM AS USED IN MUC-6," in Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995, 1995.

[16] I. B. Goodfellow, Y.; and Courville, A., Deep Learning: MIT Press, 2016.

[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," ed, 2014.

[18] B. Zhang, D. Xiong, J. Su, and M. Zhang, "Learning better discourse representation for implicit discourse relation recognition via attention networks," Neurocomput., vol. 275, pp. 1241-1249, 2018.

[19] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014, pp. 1532-1543.

[20] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," presented at the Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, Lake Tahoe, Nevada, 2013.

[21] E. F. Tjong Kim Sang, "Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition," in COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002), 2002.

[22] E. F. T. K. Sang and F. D. Meulder, "Introduction to the CoNLL-2003 shared task: language-independent named entity recognition," presented at the Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4, Edmonton, Canada, 2003.

[23] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural Architectures for Named Entity Recognition," in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, 2016, pp. 260-270.

[24] K. H. Bennett, T. M. Bull, and H. Yang, "A transformation system for maintenance: turning theory into practice," presented at the 8th International Conference on Software Maintenance, 1992.

[25] A. Katiyar and C. Cardie, "Nested Named Entity Recognition Revisited," in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), New Orleans, Louisiana, 2018, pp. 861-871.

[26] S. Pradhan, L. Ramshaw, M. Marcus, M. Palmer, R. Weischedel, and N. Xue, "CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes," in Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task, Portland, Oregon, USA, 2011, pp. 1-27.

[27] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, "CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes," in Joint Conference on EMNLP and CoNLL - Shared Task, Jeju Island, Korea, 2012, pp. 1-40.

[28] S. Wiseman, A. M. Rush, and S. M. Shieber, "Learning Global Features for Coreference Resolution," in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, 2016, pp. 994-1004.

[29] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end Neural Coreference Resolution," in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 2017, pp. 188-197.

[30] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, et al., "Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Berlin, Germany, 2016, pp. 207-212.

[31] J. Lee, S. Seo, and Y. S. Choi, "Semantic Relation Classification via Bidirectional LSTM Networks with Entity-aware Attention using Latent Entity Typing," Symmetry, vol. 11, p. 785, 2019.

[32] R. Zhang, F. Meng, Y. Zhou, and B. Liu, "Relation classification via recurrent neural network with attention and tensor layers," Big Data Mining and Analytics, vol. 1, pp. 234-244, 2018.

[33] B. Jeong, H. Cho, and C. Lee, "On the functional quality of service (FQoS) to discover and compose interoperable web services," Expert Systems with Applications, vol. 36, pp. 5411-5418, 2009.